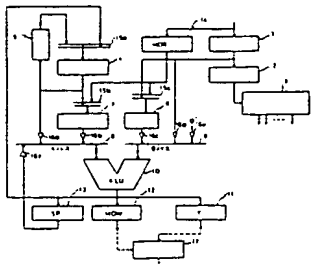
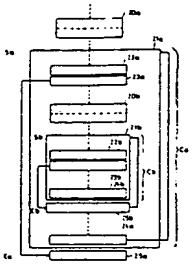
(54) INSTRUCTION CONTROL DEVICE
(11) 63-53644 (A) (43) 7.3.1988 (19) JP
(21) Appl. No. 61-197653 (22) 23.8.1986
(71) NEC CORP (72) HIROSHI SATO
(51) Int. Cl⁴. G06F9/32



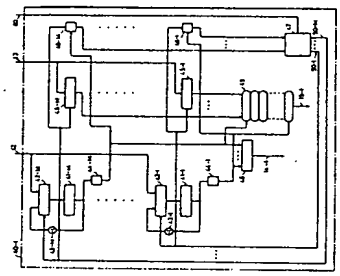


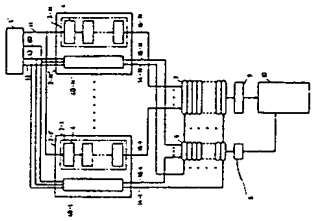
PURPOSE: To easily recognize a loop structure with a machine language instruction level, to increase an executing speed and to execute a multiplex loop by being equipped with a loop size register, a loop head address register and a system stack.

CONSTITUTION: A microprogram control part 3, when a loop starting instruction 20a is executed, sets the loop size of a loop (b) to a register 6 and sets the loop head address to the register 7 after the contents held at a loop head address register 7 and a loop size register 6 are saved onto a system stack 17. When a loop repeating instruction 24b is executed, the control is shifted to a loop head by setting the head address of the register 7 a program register 4; when a loop completing instruction 23b is executed, the contents to add the head address of the register 7 and the loop size of the register 6 are set to the register 4, and the head address and loop size of an outside loop saved at the time of starting the processing are returned to the registers 7 and 6.

1: instruction destination reading register, 2: instruction register, 5: incrementer, 22a,22b: conditions branching instruction, A,B: bud, Ca,Cb: loop size

(54) INFORMATION PROCESSING DEVICE
(11) 63-53645 (A) (43) 7.3.1988 (19) JP
(21) Appl. No. 61-197654 (22) 23.8.1986
(71) NEC CORP (72) SEIKI YOSHIDA
(51) Int. Cl⁴. G06F9/38,G06F7/00



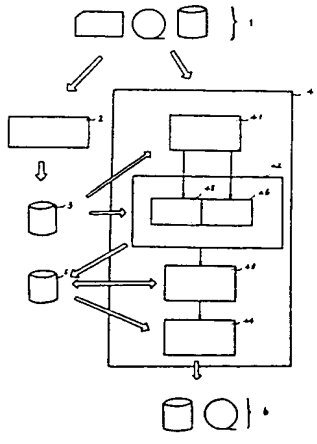


PURPOSE: To prevent the increase in the hardward quantity of an address holding circuit by adding the accommodating destination address of a register, which comes to be a pair with a counter in which a counted value comes to be a prescribed value, to a general-use register bank as a writing address.

CONSTITUTION: For a counter composed of a register 41-j, a subtracter 43-j, etc., in an address holding circuit 40-j, the executing time of an instruction is set, and counting-down is started by the staring of the execution of the instruction. Consequently, at the time of completing the execution of the instruction, a counted value comes to be a prescribed value. At this time, a control circuit 47 generates a selecting signal to selectors 6 and 7. Based on the selecting signal generated by the circuit 47, a selector 49 outputs, to a selector 6, the accommodating destination address set at a register 45-j which comes to be a pair with the counter in which a counted value comes to be a prescribed value. Consequently, the executing result of the instruction outputted from a pipeline arithmetic part 3-j is accommodated to the prescribed address of a general-use register bank 10.

1': instruction decoding part, 2-1': arithmetic unit part, 4: executing part, 8: address register, 9: data register, 42-1,42-M: changing-over circuit, 48: OR gate

(54) OPTIMUM OBJECT PROGRAM GENERATING SYSTEM
(11) 63-53646 (A) (43) 7.3.1988 (19) JP
(21) Appl. No. 61-196651 (22) 22.8.1986
(71) NEC CORP (72) AKIKAZU ABE
(51) Int. Cl⁴. G06F9/44



PURPOSE: To generating an optimum object program without getting assistance from somebody by generating the object program with a compiler based on dynamic information for optimization obtained by a dynamic information counting means.

CONSTITUTION: A dynamic information counting means 2 inputs a source program 1 assembled by high level language, counts the number of times of the execution of respective sentences and respective sub-programs included in a program, and calculates and preserves dynamic information 3 composed of the average number of rotations of respective loops and the average forming rate of respective conditional expressions. A source analyzing part 41 of a compiler 4 analyzes the program 1, refers to the information 3, executes the optimum processing and delivers a vector processing part and other part respectively to a vector processing part 45 and a non-vector processing part 46. Processing part 45 and 46 respectively refers to the information 3 and generates an intermediate text 5 to execute the object program generation with a vector instruction and a scalar instruction. For these texts 5, the result is delivered through an intermediate text optimum part 43 to an object program generating part 44 and thus, an object program 6 is generated.

42: intermediate text generating part

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　昭63－53646

⑤Int.Cl.⁴	識別記号	庁内整理番号	㊸公開　昭和63年(1988)3月7日
G 06 F 9/44	322	A－8724－5B	

審査請求　未請求　発明の数 1　（全4頁）

㊹発明の名称　最適目的プログラム生成方式

㉑特　願　昭61－196651

㉒出　願　昭61(1986)8月22日

㉒発明者　安部　晄一　東京都港区芝5丁目33番1号　日本電気株式会社内

㉑出願人　日本電気株式会社　東京都港区芝5丁目33番1号

㉔代理人　弁理士　井ノ口　壽

明　細　書

1.発明の名称

最適目的プログラム生成方式

2.特許請求の範囲

高級言語で組立てられたプログラムに含まれている各文および各サブプログラムの実行回数を計数し、各ループの平均回転数と各条件式の平均成立割合とより成る動的情報を計算して保存するための動的情報計数手段と、前記動的情報計数手段によつて得られた情報をもとに最適な目的プログラムを生成するためのコンパイラとを具備して構成したことを特徴とする最適目的プログラム生成方式。

3.発明の詳細な説明

（産業上の利用分野）

本発明は、最適な目的プログラムを生成するための処理方式に関する。

（従来の技術）

従来、コンパイラが目的プログラムを生成する際には、コンパイル時点で判明した情報だけを用いて最適化処理やベクトル化処理を行つている。従つて、コンパイラ自体では知り得ない情報をコンパイラに知らせるためにはプログラムのなかに指示行を挿入するか、あるいはコンパイル時に直接利用者が指示を与える手段が必要であつた。

（発明が解決しようとする問題点）

上述した従来の最適目的プログラム生成方式においては、利用者自身がプログラムのなかのループの実行回数や、条件式の成立割合を正確に把握しておかなければならず、それらの情報を利用者自身がコンパイラに与えなければならないと云う欠点がある。

また、プログラムが固定されていない場合には、プログラム修正の都度、上記の情報を人手を介して与えなおさなければならないと云う欠点もある。

本発明の目的は、ベクトル演算手段を備え、コンパイラ方式の高級言語で組まれたプログラ

ムのなかに含まれている各文、および各サブプログラムの実行回数を計数し、各ループの平均回転数や各条件式の平均成立割合などの動的情報を計算して保存し、上記プログラムのコンパイル時に、得られた情報をもとにして最適な目的プログラムをコンパイラにより生成することにより上記欠点を除去し、人手の介入なくプログラムを生成できるように構成した最適目的プログラム生成方式を提供することにある。

（問題点を解決するための手段）

本発明による最適目的プログラム生成方式は動的情報計数手段と、コンパイラとを具備して構成したものである。

動的情報計数手段は、高級言語で組立てられたプログラムに含まれている各文および各サブプログラムの実行回数を計数し、各ループの平均回転数と各条件式の平均成立割合とより成る動的情報を計算して保存するためのものである。

コンパイラは、動的情報計数手段によつて得られた情報をもとに最適な目的プログラムを生

成するためのものである。

（実　施　例）

次に、本発明について図面を参照して説明する。

第１図は、本発明による最適目的プログラム生成方式を実現する一実施例を示すブロック図である。第１図において、１はソースプログラム、２は動的情報計数手段、３は動的情報、４はコンパイラ、４１はソース解析部、４２は中間テキスト生成部、４３は中間テキスト最適化部、４４は目的プログラム生成部、４５はベクトル化処理部、４６は非ベクトル化処理部、５は中間テキスト、６は目的プログラムである。

動的情報計数手段２はコンパイラ方式の高級言語で組まれたソースプログラム１を入力し、当該プログラムのなかの文がコンピュータで走行時に実行されるごとに、この文の実行回数を積算することによつてプログラム１の文ごとの実行回数を得ると同時に、プログラム１を構成する副プログラムごとの実行回数をも積算する

ことによつて、最終的にはプログラム１の各ループの平均回転数や、各条件式の平均成立割合のコンパイラが最適な目的プログラムを生成するために有用な動的情報３を計算して保存するプログラムである。

コンパイラ４の内部のソース解析部４１は供給されたソースプログラム１を解析し、ベクトル化処理部分をベクトル化処理部４５に渡し、他の部分を非ベクトル化処理部４６に渡す。この際、ソース解析部４１は動的情報計数手段２によつて保存された動的情報３を参照し、例えば第２図に示すようにＦＯＲＴＲＡＮのＤＯ文によつて構成されるループの平均回転数が少なくて、ベクトル命令列で実行するよりもスカラ命令列で実行した方が効率がよいと判断される場合にはＤＯ文、およびＤＯ文の制御範囲の文を非ベクトル化処理部４６に渡す。中間テキスト生成部４２の内部のベクトル化処理部４５はソース解析部４１から渡されたプログラムに対し、ベクトル命令を用いた目的プログラムを生成す

るための中間テキスト５を生成する。

いつぽう、非ベクトル化処理部４６は、スカラ命令を用いた目的プログラムを生成するための中間テキスト５を生成する。この際にも、動的情報３を参照し、最適化が行われる。例えば第３図に示すように、ＦＯＲＴＲＡＮのＩＦ文の条件式が成立する割合に応じて、最適な中間テキスト５が生成される。

上述のような動的情報３の参照、考慮によつて生成された中間テキスト５は、中間テキスト最適化部４３において周知の最適化処理が実行されたのち、目的プログラム生成部４４に結果が渡され、これによつて目的プログラム６が生成される。

第１図に示す実施例では、ソース解析部４１および中間テキスト生成部４２において動的情報３を参照し、最適な中間テキストを生成しているが、中間テキスト最適化部４３において動的情報３を参照し、最適な中間テキストに書換える構成としてもよい。また、動的情報３の参

照はソース解析部４１だけにとどめ、コンパイラの内部テーブルに上記情報を保持しておき、中間テキスト生成部４２において上記テーブルを参照してもよい。

（発明の効果）

以上説明したように本発明によれば、ベクトル演算手段を備え、コンパイラ方式の高級言語で組まれたプログラムのなかに含まれている各文、および各サブプログラムの実行回数を計数し、各ループの平均回転数や各条件式の平均成立割合などの動的情報を計算して保存し、上記プログラムのコンパイル時に、得られた情報をもとにして最適な目的プログラムをコンパイラにより生成することにより、利用者自身がコンパイラに直接指示を与えることなく、最適化のための正確な情報をコンパイラに与えることができるので、目的プログラムの実行性能が向上すると云う効果がある。

4.図面の簡単な説明

第１図は、本発明による最適目的プログラム生成方式の一実施例を示すブロック図である。

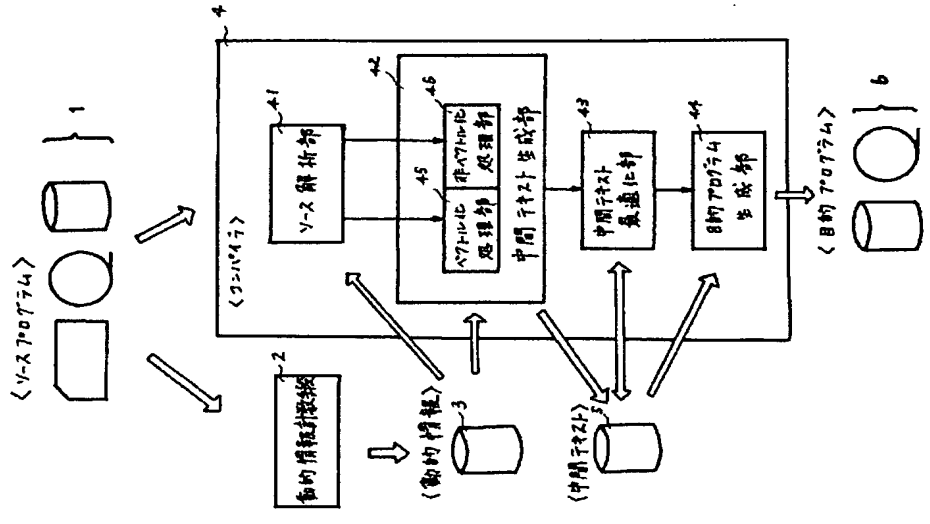
第２図は、ＦＯＲＴＲＡＮのＤＯ文に対する最適化例を示す説明図である。

第３図は、ＦＯＲＴＲＡＮのＩＦ文に対する最適化例を示す説明図である。

１…ソースプログラム
２…動的情報計数手段　３…動的情報
４…コンパイラ　５…中間テキスト
６…目的プログラム　４１…ソース解析部
４２…中間テキスト生成部
４３…中間テキスト最適化部
４４…目的プログラム生成部
４５…ベクトル化処理部
４６…非ベクトル化処理部

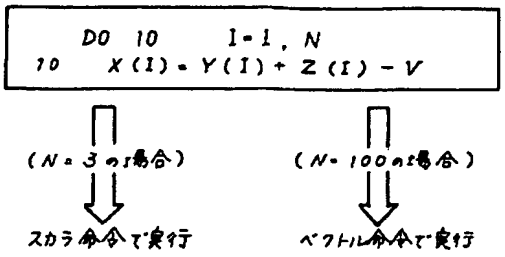
特許出願人　日本電気株式会社

代理人　弁理士　井　ノ　口　　壽

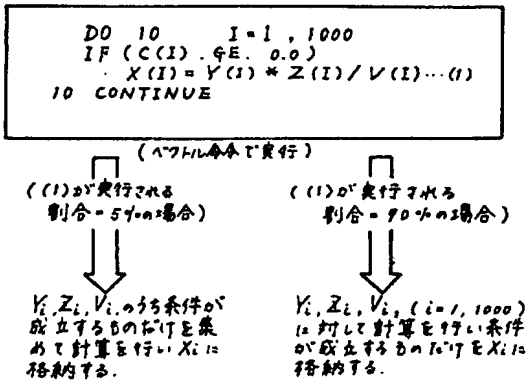


第１図

第2図


DO 10 I=1, N
10 X(I)=Y(I)+Z(I)-V
(N=3の場合)
スカラ命令で実行
(N=100の場合)
ベクトル命令で実行

第3図


DO 10 I=1, 1000
IF (C(I).GE. 0.0)
X(I)=Y(I)*Z(I)/V(I)…(1)
10 CONTINUE
(ベクトル命令で実行)
((1)が実行される割合=5%の場合)
Yi, Zi, Viのうち条件が成立するものだけを集めて計算を行いXiに格納する.
((1)が実行される割合=90%の場合)
Yi, Zi, Vi, (i=1, 1000)に対して計算を行い条件が成立するものだけをXiに格納する.